LINKgroup

Distance Learning System

PyTorch

Biblioteka PyTorch

Python Data Access and Processing

Šta je PyTorch

- PyTorch je biblioteka za rad sa višedimenzionalnim nizovima (tenzorima)
- PyTorch se najčešće koristi za kreiranje neuronskih mreža

pip install torch torchvision

Kreiranje tenzora

```
arr = torch.tensor(
    [1,2,3,4]
)
```

Automatski tipizirani tenzor

```
arr = torch.tensor(
    [1,2,3,4],
    dtype=torch.float
)
```

Eksplicitno tipizirani tenzor

```
arr = torch.from_numpy(
    numpy.array([1,2,3,4])
)
```

Tenzor od numpy niza

```
arr = torch.tensor(
    [[1,2,3,4],[2,3,4,5],[3,4,5,6]]
)
```

"Dvodimenzionalni" tenzor

Aritmetičke operacije nad tenzorima

- Većina operacija biblioteke numpy primenjiva je i na Pytorch tenzore

- Operacije:

  + , - , * , /

- Metode:

  t1.add(t2)

  t1.dot(t2)

  t1.matmul(t2)

Generisanje i preoblikovanje tenzora

- Većina operacija biblioteke numpy primenjiva je i na Pytorch tenzore

`torch.zeros(3,3)`

```
0 0 0
0 0 0
0 0 0
```

`torch.rand(3,2)`

```
0.68 0.39
0.73 0.68
0.60 0.69
```

`torch.eye(3,3)`

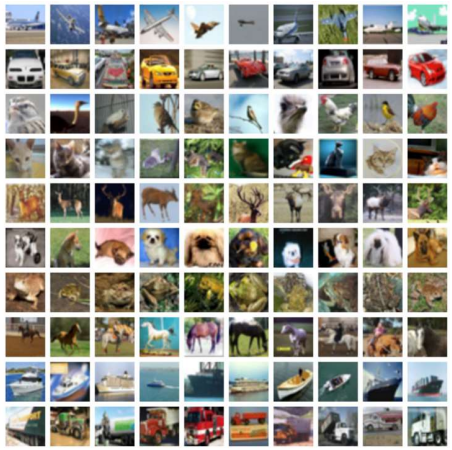
```
1 0 0
0 1 0
0 0 1
```

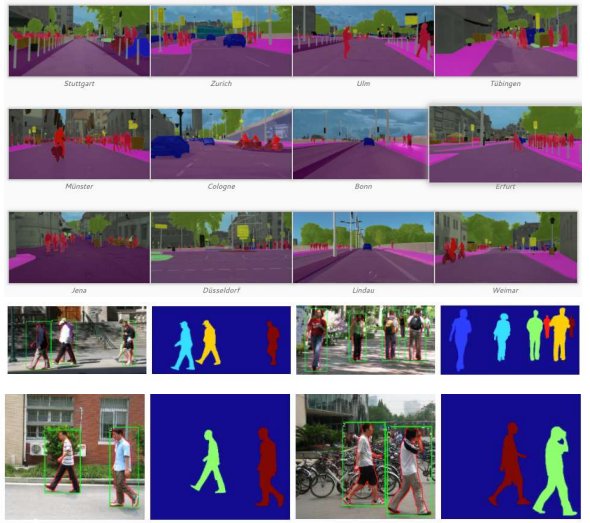
`torch.reshape(2,3)`

```
0.68 0.39 0.73
0.68 0.60 0.69
```

Pytorch setovi podataka

- Pytorch omogućava preuzimanje mnoštva setova podataka i predefinisanih modela pomoću paketa **torchvision**





Aktivacione funkcije

```
sig = nn.Sigmoid()
print(
    sig(
        torch.tensor([5.0])
    )
)
```

```
rel = nn.ReLU()
print(
    rel(
        torch.tensor([-5.0])
    )
)
```

```
smax = nn.Softmax()
print(
    smax(
        torch.tensor([1.0,2.0,3.0])
    )
)
```

Loss funkcije

```
loss_f = nn.CrossEntropyLoss()
target = torch.tensor([0])
pred = torch.tensor([[1.0,0.5,0.1]])
loss = loss_f(pred,target)
```

0.6997

```
loss_f = nn.MSELoss()
target = torch.tensor([1.0,4.0,9.0])
pred = torch.tensor([[1.0,2.0,3.0]])
loss = loss_f(pred,target)
```

13.3333

PyTorch model

1 ulaz

Jedan sloj modela

1 izlaz

```
model = nn.Linear(1,1)
```

```
loss_f = nn.MSELoss()
```

Loss funkcija

Optimizator
(stochastic gradient descent)

Težine

Learning rate

```
opt = torch.optim.SGD(params=model.parameters(),lr=0.01)
```

Forward propagation

Loss funkcija

Back propagation

Ažuriranje težina

Reset optimizatora

```
pred = model(data)
loss = loss_f(target,pred)
loss.backward()
opt.step()
opt.zero_grad()
```

Proširivanje modela

- Svaki model baziran je na klasi Module
- Ovaj model se nasleđuje za realizaciju kompleksnijih mreža

```
class MyNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear = nn.Linear(2,1)
    def forward(self,x):
        return torch.sigmoid(self.linear(x))
```

Duboka mreža

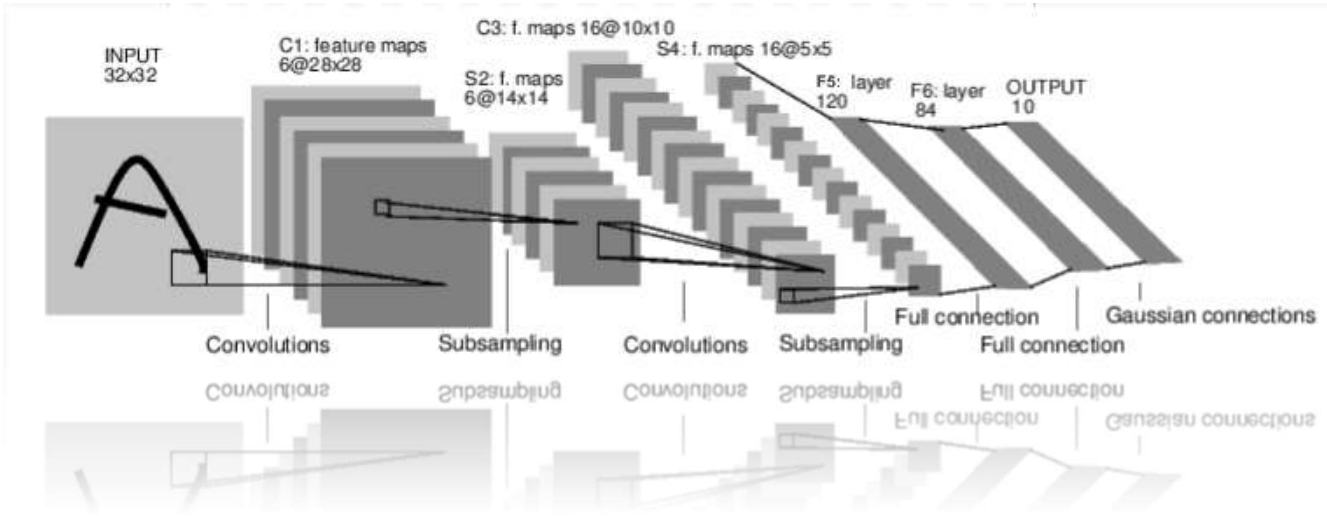
- Jedan model može sadržati više modela, čime se formira duboka mreža

```
class MyModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.lin1   = nn.Linear(72*72*3,100)
        self.relu    = nn.ReLU()
        self.lin2   = nn.Linear(100,12)
    def forward(self, x):
        out = self.lin1(x)
        out = self.relu(out)
        out = self.lin2(out)
        return out
```

Konvolucionalna mreža

(pdap-ex02 faces)

- Za rad sa slikama (prepoznavanje lica ili oblika) najčešće se koristi konvolucionalna mreža



Preuzimanje dataset-a

- Pytorch omogućava preuzimanje mnoštva setova podataka i predefinisanih modela pomoću paketa **torchvision**

```
dataset = torchvision.datasets.MNIST(
    root="./data",
    download=True
)
```

Credits

https://www.flaticon.com/authors/freepik https://www.flaticon.com/authors/nikita-golubev